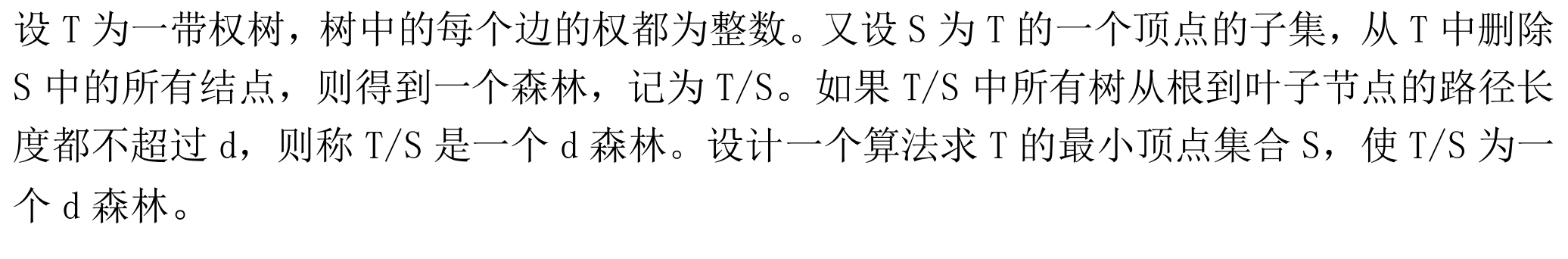
西安交通大学实验报告

课程名称：算法设计与分析 实验名称： 分治与递归

学 院：\_\_\_\_\_\_\_电信学部\_\_\_\_\_\_\_\_\_\_\_\_ 实 验 日 期 2022年4 月 24 日

|  |  |  |
| --- | --- | --- |
| **刘沁宇** | **计算机002** | **2203613019** |

#### 问题描述



#### 问题分析

由于设计到树的问题，因此需要采取合适的数据结构存储相应的信息。且要保证删除结点数最小的话可采取树的自下而上的BFS遍历方法，确保考虑到每种情况。

#### 算法设计

利用父亲列表的形式来存储每个结点的信息，具体如下。

|  |
| --- |
| 1. **struct** node         //结点信息 2. { 3. **int** parent;    //父亲结点序号 4. **int** out;       //出度 5. **int** len;       //到父亲结点的距离 6. **int** maxl;      //到叶子结点的最大距离 7. **bool** cut;      //判断是否已经被删除 8. }; 9. **class** dTree 10. { 11. **private**: 12. queue <node> Q;      //用队列实现BFS 13. vector <node> tr;    //结点序列 14. **int** maxlen;          //最大长度d 15. **int** sum;             //删除的结点数 16. **public**: 17. dTree(**int** n, **int** d); 18. **void** solution(); 19. } |

若当前节点不是根节点,则向上移动,路长超过d且未分离时将结点删除并向根节点移动;若没有超过d则更新maxl。若出度为0则将该结点添加到队列Q中，先调用析构函数并建立树,再调用solution求解。

#### 算法实现

|  |
| --- |
| 1. #include <iostream> 2. #include <fstream> 3. #include <string> 4. #include <cmath> 5. #include <vector> 6. #include <algorithm> 7. #include <queue> 8. **using** **namespace** std; 9. // 算法要求：设计一个算法求 T 的最小顶点集合 S，使 T/S 为一个 d 森林。 10. **struct** node         //结点信息 11. { 12. **int** parent;    //父亲结点序号 13. **int** out;       //出度 14. **int** len;       //到父亲结点的距离 15. **int** maxl;      //到叶子结点的最大距离 16. **bool** cut;      //判断是否已经被删除 17. }; 18. **class** dTree 19. { 20. **private**: 21. queue <node> Q;      //用队列实现BFS 22. vector <node> tr;    //结点序列 23. **int** maxlen;          //最大长度d 24. **int** sum;             //删除的结点数 25. **public**: 26. dTree(**int** n, **int** d) 27. { 28. maxlen = d; 29. tr.resize(n); 30. tr[0].parent = -1; 31. **for** (**int** i = 0; i < n; i++) 32. { 33. **int** size, length, num; 34. cin >> size; 35. tr[i].maxl = 0; 36. tr[i].cut = **false**; 37. tr[i].out = size; 38. **for** (**int** j = 0; j < size; j++) 39. { 40. cin >> num >> length; 41. tr[num].parent = i; 42. tr[num].len = length; 43. } 44. } 45. sum = 0; 46. } 47. **void** solution() 48. { 49. **for** (unsigned **int** i = 0; i != tr.size(); i++) 50. { 51. **if** (tr[i].out == 0) 52. Q.push(tr[i]); 53. } 54. **while** (!Q.empty()) 55. { 56. node temp = Q.front(); 57. Q.pop(); 58. **int** par = temp.parent; 59. **if** (tr[par].cut == **false** && temp.len + temp.maxl > maxlen) 60. { 61. tr[par].cut = **true**; 62. sum++; 63. par = tr[par].parent;     //此时要将par指向被减去结点的父亲结点 64. } 65. **else** **if** (tr[par].cut == **false**) 66. { 67. **if** (tr[par].maxl < temp.len + temp.maxl) 68. tr[par].maxl = temp.len + temp.maxl; 69. } 70. **if** (par > 0 && --tr[par].out == 0) 71. Q.push(tr[par]); 72. } 73. cout << sum; 74. } 75. }; 77. **int** main() { 78. **int** n, d;               //n为顶点个数，d为路径长度 79. cin >> n >> d; 80. dTree dt(n, d);    //构建与初始化树 81. dt.solution();      //通过solution函数输出结果 82. **return** 0; 83. } |

#### 实验结果

